REMARKS

Entry of this Amendment under 37 CFR §1.116 is respectfully requested. Claims 1-26 and 34-35 are pending in the application.

The foregoing amendments to claims 7, 13, 16-17, 20, 24, and 35 place the claims in better condition for appeal. Specifically, the recital of "one of [A] and [B]" has been replaced with the recital of "one of [A] *or* [B]" to ensure the claims are accurately interpreted that either A *or* B should be used, as opposed to the improper interpretation that *both* A *and* B must be used, as suggested recently by the Federal Circuit.[1]

Use of the term "or" has been deemed acceptable under 35 USC §112, second paragraph. *See* MPEP 2173.05(h)II. at 2100-222 (Rev. 3, Aug. 2005) (*citing In re Gaubert*, 524 F.2d 1222, 187 USPQ 664 (CCPA 1975)).

Further, the interpretation of "one of [A] and [B]" has never been an issue during prosecution, and does not affect claim scope with respect to the outstanding §103 rejections. Hence, it is believed entry of this Amendment is proper because it does not raise any new issues or require a new search.

Claims 1-26, 34, and 35 were rejected under 35 USC §103 in view of U.S. Patent Publication No. 2002/0023147 by Kovacs in view of U.S. Patent Publication 2004/0210635 by Raniere and US Patent No. 5,926,177 to Hatanaka et al. This rejection is respectfully traversed, as the rejection fails to establish a prima facie case of obviousness, but rather presents assertions that are not only unfounded, but which *disregard* the explicit claim language and *contradict* the explicit teachings of the references.

As demonstrated below, the rejection fails to demonstrate that the hypothetical combination discloses or suggests transfer of a service object (e.g., a model object, a view object, or a controller object) associated with a network service, as specified in independent claim 1, 7, 13, 20, or 35. Moreover, the final rejection is replete with inconsistencies, unfounded assertions,

---

[1] *See Superguide Corp v. DirecTV Enterprises, Inc.*, 358 F.3d 870, 886-87 (Fed. Cir. 2004).

and gross mischaracterizations of the applied references that demonstrate a hindsight reconstruction that is legally impermissible.

Kovacs et al.

The rejection states at page 6 that the Examiner disagrees with the assertion that Kovacs fails to teach execution of service objects in a user interface device. Rather than providing a citation to Kovacs et al. that unequivocally demonstrates that Kovacs et al. teaches execution of service objects in a user interface device, the rejection presents a blatant mischaracterization of Kovacs et al. by asserting that "[b]ecause the view object can be manipulated by the client for example, the client necessarily contains the service object."

In particular, the rejection fails to demonstrate that the client "necessarily contains the service object" simply because the view object can be manipulated by the client. In fact, this statement is inconsistent with the explicit description of Kovacs et al., which describes that the client device includes a browser 11 that sends a request (1) to the controller servlet 13 of the network service: Figure 1 illustrates that a client 11 with a browser can access a multimedia service 1 including a portal 5 (paragraph 38); Figures 2-4 explicitly illustrate that the browser 11 is distinct from the network service, and that the browser 11 sends a request (1) to the controller object of the service, and that the view object 12 of the service sends the response (5) to the browser 11; paragraph 39 explicitly specifies with respect to Figure 2 that "these services are all constructed according to the MVC (model-view-controller) architecture and comprise **a controller 13 being a servlet corresponding to a small program run on a server**, a view 12 for the presentation via mark-up languages and a model 14 containing data"; paragraph 44, lines 1-5 describe with respect to Figure 1 that the request **from the browser** is issued **to the controller servlet**, and paragraphs 84 and 87 describe with respect to Figure 5 that the portal services controller receives a request **from the user agent browser**. Finally, all of the claims 1-10 of Kovacs et al. specify a portal application for providing access "**from a client (11) to a multimedia service (1)**", wherein the portal application includes a plurality of services structured according to the model-view-controller architecture.

Hence, the statement in the Final Action that "the client necessarily contains the service object" is not only unfounded, but is refuted by Kovacs et al. which *consistently* teaches that **none** of the model, view, or controller objects are implemented in the client device 11 containing the browser; rather, Kovacs et al. consistently teaches that the browser in the client device 11 sends a **request** to the controller implemented as a servlet in the server. As illustrated in the attached Exhibit A, a "servlet" is a program that is executed on a server, as opposed to Java "applets", which are intended for execution on a client.

In addition, the statement on page 6 that "execution of a service object may be constituted by GUI inputs" fails to recognize that Kovacs et al. teaches that the request that is output by the client device 11 is received by the controller implemented as a servlet in the server. The mere reception of a GUI input has not been demonstrated to be a teaching of a claimed service object.

Hence, the Final Action fails to rebut applicants argument that Kovacs et al. fails to teach execution of service objects in a user interface device. For this reason alone the 103 rejection must be withdrawn.

Raniere et al.

The statement in the Final Action that "[a] broad reasonable interpretation of a MVC paradigm is constituted in the viewing of HTML pages, where the display is the view" is not only lacking in foundation, but also demonstrates a disregard for the explicit claim language; moreover, the supposed "broad reasonable interpretation" is inconsistent with both the specification and the interpretation one having ordinary skill in the art would reach, as demonstrated by Kovacs et al. in Hatanaka et al..[2]

---

[2]"During patent examination, the pending claims must be 'given their broadest reasonable interpretation consistent with the specification.'" MPEP §2111 at 2100-46 (Rev. 3, Aug. 2005) (*quoting In re Hyatt*, 211 F.3d 1367, 1372, 54 USPQ2d 1664, 1667 (Fed. Cir. 2000)).

"The broadest reasonable interpretation of the claims must also be consistent with the interpretation that those skilled in the art would reach." MPEP §2111.01 at 2100-47 (Rev. 3, Aug. 2005) (*citing In re Cortright*, 165 F.3d 1353, 1359, 49 USPQ2d 1464, 1468 (Fed. Cir.

Moreover, the supposed "broad reasonable interpretation" does not support the premise that "given that Raniere deals in the transferring of objects, which may obviously include HTML pages, display features, or other data, it fits into the field of an MVC paradigm", because the claims that specify transfer of a ***service object***, as opposed to simply a data object.

The Final Action disregards the explicit claim language that "the first network service" supplied to the user based on "exchange of service transaction messages between a corresponding group of service objects including a model object, a view object, and a controller object associated with the first network service", as specified in claims 1, 7, 13, 20, and 35. Specifically, the claimed "first network service" is executed based on exchange of service transaction messages between a model object, a view object, and a controller object, where each of the service objects are executed to provide the first network service.

In other words, the broadest reasonable interpretation requires that each service object, including each of the model object, view object, and controller object, be executed in order to provide the network service, hence a mere "data object" as disclosed in Raniere et al. cannot be considered a teaching of the claimed ***service*** object.

For example, the specification consistently describes that the service object perform operations based on received data (e.g., the received data being an XML page):

> The view object is configured for ***mapping (i.e., rendering) data*** such as text and/or graphics onto a device. Changes to a model object causes the view object to automatically redraw the affected part of the image to reflect the changes. In addition, multiple view objects for the same model may be utilized to ***render*** the contents of the model object according to a different corresponding display (e.g., data table, monthly chart view, daily chart view, linear scale, logarithmic scale, etc).
>
> (Page 3, lines 6-10).

The specification also describes that "output display ***operations [are] performed by a view object***" (page 7, lines 24):

1999)).

The client device 10 includes an input controller resource 18 configured as a controller object for outputting a request based on a user input, and ***a display controller 20 configured as a view object for displaying data according to prescribed formats***. Hence, the controller object 18 sends service transaction message specifying a request to the model object 16; the model object 16 includes predefined routines (i.e., logic and data) enabling the retrieval of the requested data; *the data is sent in a service transaction message output by the model object 16 to the view object 20, for example in the form of data table or an XML document.* ***The display controller 20, serving as the view object, displays the stock quote data in a prescribed format***, for example a table or a graph.

(Page 8, line 27 to page 9, line 6)

As described further below, model objects 16 can be executed by the application controller 128 (e.g., by the services manager 130c), view objects 20 can be executed by the XML module 38, and the controller objects 18 can be controlled by the message controller 35 using input definitions stored in the XML file store 40.

(page 14, lines 17-20).

Hence, the specification describes that the view object is not simply the "data" that is to be displayed (as suggested by the Examiner), rather the view object is an executable resource that performs the operation of *displaying the data.*

Further, Kovacs et al. explicitly specifies that implementation of any one of the model object, view object, or controller object requires execution of software: Kovacs et al. explicitly specifies that a view object is executable code configured for the presentation of information *using* markup languages (i.e., executing the markup languages): "The model-view-controller architectural pattern divides an **interactive application** into three components. The model contains the core functionality data. Views display information to the user. Controllers handle user input." (Para. 40). "**The view object can be f.e. [sic] a Java Server Page (JSP)**. JSP pages allow easily mixing presentation (e.g. HTML elements) with service logic in the form of Java bean objects. The Java bean object accesses the model object for retrieving data or for executing the service logic." (Para. 44) "This view object does not have to care for these details, but **prepares the information page that is displayed to the user**. Through the JSP concept,

mark-up language and service logic is clearly separated." (Para. 48).

> [0083] One controller 13 of a service can be associated with a plurality views 12, 12', 12" for representing the same data of the model component 14 with different mark-up languages. In other words, **each of the views 12, 12', 12" is responsible for the representation of a data of the model component 14 with a particular mark-up language**.

Hence, even Kovacs et al. explicitly specifies that the view object is implemented as executable code.

Hatanaka et al. *also* requires that the view object performs operations, and cannot be as broadly construed as a "data object"(col. 6, lines 2-7 and 52-63). Even Hatanaka et al. teaches that all of the model view and controller objects are part of a single executable program (e.g., column 3, lines 12 and 27), where the single model and the view proxy each are connected to the controller (abstract, claim 1).

Hence, there is no rational basis to interpret the claimed "service object" (expressed as being one of a model object, a view object, or a controller object) for a "network service" as broad enough to encompass the data objects disclosed in Raniere et al., admitted by the Examiner as being limited to HTML pages or other data.

Moreover, the Final Action fails to demonstrate that Raniere et al. is analogous art: in fact, there is absolutely no reference whatsoever in Raniere et al. of the phrase "MVC" or "model view controller" or any equivalent, and the only reference to the term "model" is in paragraph 76, which refers to a "client/server model". In fact, Raniere et al. explicitly describes that the "data objects"are actually data files (specified in paragraph 34) that are retrieved by users before a telephone conference begins (see, e.g., paragraphs 4, 5, 33, and 37). Raniere et al. provides no description whatsoever of even transferring executable code, let alone any model, view, or controller objects, but simply describes users retrieving data objects.

As noted by the MPEP § 2141.01 (a), page 2100-119, the differences in structure and function of the inventions carry far greater weight in determining that a reference is not

analogous art. Raniere et al. teaches that users retrieve data *before* initiating a telephone conference, and provides no description of an application service that is partitioned into the claimed model object, view object, or controller object.

Finally, the assertion that Raniere et al. "may therefore be combined with the other references to arrive at the claimed invention" demonstrates a disregard of the legal standard that requires evidence of a desirability to modify Kovacs et al. in the manner claimed. "Teachings of references can be combined only if there is some suggestion or incentive to do so." *In re Fine*, 5 USPQ2d 1596,1600 (Fed. Cir. 1988) (*quoting ACS Hosp. Sys. v. Montefiore Hosp.*, 221 USPQ 929, 933 (Fed. Cir. 1984)) (emphasis in original). The supposed motivation of "moving objects between users would allow sharing" fails to address the claimed feature of moving ***service objects.***

Modifying Kovacs et al. with Raniere et al.

In addition to the foregoing, *even assuming* Raniere et al. taught transferring a "view object", as asserted in the Final Action, the hypothetical modification of Kovacs et al. with Raniere et al. would be improper because it would render Kovacs et al. unsatisfactory for its intended purpose[3], and destroy the invention of claims 3-5 in Kovacs et al.,[4] since the controller

---

[3]The proposed modification cannot change the principle operation of a reference or render it unsatisfactory for its intended purpose. "If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims prima facie obvious." MPEP § 2143.02, Rev. 3, Aug. 2005 at p. 2100-138 (Citing In re Ratti, 270 F.2d 810, 123 USPQ 349 (CCPA 1959). "If the proposed modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification." Id. at 2100-137 (Citing In re Gordon, 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984)). Cf. MPEP §2145.III at page 2100-167 (Rev. 3, Aug. 2005) ("the claimed combination cannot change the principle of operation of the primary reference or render the reference inoperable for its intended purpose.").

[4]*See Ex parte Hartmann*, 186 U.S.P.Q. 366, 367 (USPTO Bd. Pat. App. 1974) (reversing rejection when modification would destroy basis for invention in one or two references).

component in Kovacs et al. would no longer be able to control the plurality of views for different presentations. Specifically, Kovacs et al. describes the view object as a Java server page: Java server pages are known as programs specified in a webpage and that are executed on the Web server to modify the webpage before it is sent to the user who requested it (see attached Exhibit B). Hence, modifying Kovacs et al. would frustrate the purpose as described in paragraph 53, where different views can be dynamically selected through the controller to make it easier to provide different output for different devices and markup languages.

Further, the proposed modification would destroy the invention of claims 3-5 in Kovacs et al., which explicitly specifies that the portal application provides access from a client **to a multimedia service**, wherein the portal application comprises a plurality of services structured according to the model-view-controller architecture where the controller component of a service is designed to control a plurality of views component for different presentations, and where the controller is designed to select one of the views component based on the browser characteristic of the client. In other words, the proposed modification would destroy the invention of claims 3-5 because the claimed controller component would no longer be able to control the views component for different presentations, because the proposed modification would move the views component from the server to the client device, preventing the controller from being able to control or select a view as claimed.

For these and other reasons, the §103 rejection should be withdrawn.

The Hypothetical Combination of Kovacs et al., Raniere et al., and Hatanaka et al.

One skilled in the art, upon reviewing Kovacs et al., Raniere et al., and Hatanaka et al., would not have been motivated to create the claimed feature of ***transferring*** a ***service object*** between network nodes, as claimed, because both Kovacs et al. and Hatanaka et al. consistently teach that the model object, the view object, and the controller object are static and **do not move**. Kovacs et al., described above, shows that each of the model object, the view object, and the controller object reside in the **server**; Even Hatanaka et al. teaches that all of the model view and controller objects are part of a **single executable program** (e.g., column 3, lines 12 and 27),

where the single model and the view proxy each are connected to the controller (abstract, claim 1).

Further, Hatanaka et al. teaches use of a view proxy in order to **switch between different views** (e.g., col. 2, lines 48-59; col. 3, lines 47-50; col. 4, lines 4-17), where all of the model view and controller objects are implemented within a single program; moreover, Hatanaka et al. does not teach the selective termination of an object based on *receiving another object*, as claimed, but teaches that a **user** may decide to use another new object, **resulting in the *creation* of the new view object without even terminating the old view object**:

> The second approach has been to **replace** the View object in the Model-View-Controller configuration **as the type of information the user is interested in changes**. ***When the user is no longer interested in the structure of the distributed application and becomes interested in the detailed state of the components, the structure view is replaced with a detailed state view***. The problem with this approach is that **the new view must be recreated on demand** and inserted into the Model-View-Controller configuration. ***Even if the old view is preserved when the user switches out of a particular mode*** (switching from structure to state views, for example), there is expense involved when the old view is reused (switching back to structure from state). The old view must be reconnected with the model and controller, and it must be made current to reflect any model changes that occurred when it was not active. This expense is enough to inhibit the user from switching between views freely.
>
> (Col. 2, lines 1-10)

Hence, Hatanaka et al. does not teach or suggest the claimed "terminating the received one service object ***based on reception, via the network interface and the open protocol network, of a second service object***". Rather, Hatanaka et al. teaches that the user device creates a new view object in response to a user selection for a new view, and teaches that the old view may still be preserved!

Hence, there is no disclosure or suggestion that "a new view object *arrives*", as asserted. Rather, a new view object is created in response to a user selection. "The mere fact that the prior art may be modified in the manner suggested by the Examiner does not make the modification obvious unless the prior art suggested the desirability of the modification." *In re Fritch*, 23 USPQ2d 1780, 1783-84 (Fed. Cir. 1992). *In re Mills*, 16 USPQ2d 1430 (Fed. Cir. 1990).

For these and other reasons, the rejection of independent claims 1, 7, 13, 20, and 35 must be withdrawn.

Independent Claims 13 and 20

The §103 rejection of independent claims 13 and 20 also is legally deficient because it fails to address each and every claim limitation. Specifically, the Final Action states on page 5 that "claims 13-26 are rejected on the same bases [sic] as claims 1-11".

However, independent claims 13 and 20 specify the claimed feature of "transferring a selected service object via the open protocol network and between any one of the service node, the network-enabled user interface device, or a second network node based on a prescribed condition ***and while maintaining a user-perceived continuous service of the first network service***." The Final Action fails to address this claimed feature, and therefore is *per se* deficient.[5]

Further, as demonstrated above, Raniere et al. requires all data objects to be retrieved by clients before a conference call begins in order to avoid interruptions in the voice conversations (para. 4, 5, 33, 37).

For these and other reasons, the §103 rejection of independent claims 13 and 20 must be withdrawn.

Dependent Claims 3 and 9

Applicant further traverses the rejection of claims 3 and 9, which specify that the user interface device provides the first network service "based ***solely on execution of the controller***

---

[5]It is well settled that each and every claim limitation must be considered. As specified in MPEP §2143.03, entitled "**All Claim Limitations Must Be Taught or Suggested**": "To establish prima facie obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). 'All words in a claim must be considered in judging the patentability of that claim against the prior art.' *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970)." MPEP §2143.03 at 2100-139 (Rev. 3, Aug. 2005).

object, wherein the model object and the view object are executed remotely relative to the device." The rejection of claims 3 and 9 is inconsistent with the arguments presented with respect to claims 1 and 7, where the Examiner asserted that "Raniere teaches the transmitting of **view objects** through a network in response to user input (paragraph 33)." Hence, the Examiner's arguments with respect to the teachings of Raniere are inconsistent between the independent claims 1 and 7 compared to dependent claims 3 and 9.

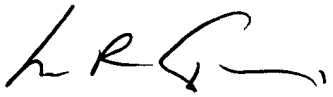
Further, para. 42 of Kovacs et al. is part of a summary description of model-view-controller technology (beginning at para. 40 and concluding at para. 42): para. 43 explicitly specifies that "[b]ased on these technologies, a service can be build [sic] ... through the utilisation [sic] of the model-view-controller pattern." The piecemeal application of Kovacs et al. is improper: the reference must be considered in its entirety, i.e., as a whole, including portions that would lead away from the claimed invention (see MPEP 2141.02 at page 2100-95 (Rev. 1, Feb. 2000) (citing *W.L. Gore & Associates, Inc. v. Garlock, Inc.*, 22 USPQ 303 (Fed. Cir. 1983), *cert. denied*, 469 U.S. 851 (1984))).

For these and other reasons, the rejection of claims 3 and 9 must be withdrawn.

In view of the above, it is believed this application is in condition for allowance, and such a Notice is respectfully solicited.

To the extent necessary, Applicant petitions for an extension of time under 37 C.F.R. 1.136. Please charge any shortage in fees due in connection with the filing of this paper, including any missing or insufficient fees under 37 C.F.R. 1.17(a), to Deposit Account No. 50-1130, under Order No. 95-468, and please credit any excess fees to such deposit account.

Respectfully submitted,



Leon R. Turkevich
Registration No. 34,035

Customer No. 23164
Date: October 24, 2006

servlet

SearchWebServices.com Definitions

EMAIL THIS

(Powered by WhatIs.com)

LOOK UP TECH TERMS

Powered by: WhatIs.com

Search listings for thousands of IT terms:

SEARCH

Browse tech terms alphabetically:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #

servlet

DEFINITION - A servlet is a small program that runs on a server. The term was coined in the context of the Java applet, a small program that is sent as a separate file along with a Web (HTML) page. Java applets, usually intended for running on a client, can result in such services as performing a calculation for a user or positioning an image based on user interaction.

Some programs, often those that access databases based on user input, need to be on the server. Typically, these have been implemented using a Common Gateway Interface (CGI) application. However, with a Java running in the server, such programs can be implemented with the Java programming language. The advantage of a Java servlet on servers with lots of traffic is that they can execute more quickly than CGI applications. Rather than causing a separate program process to be created, each user request is invoked as a thread in a single daemon process, meaning that the amount of system overhead for each request is slight.

Instead of a URL that designates the name of a CGI application (in a "cgi-bin" subdirectory), a request in a form on a Web HTML page that results in a Java servlet getting called would call a URL that looks like this:

```
http://www.whatis.com:8080/servlet/gotoUrl?http://www.someplace.com
```

The "8080" port number in the URL means the request is intended directly for the Web server itself. The "servlet" would indicate to the Web server that a servlet was being requested.

Add-on modules allow Java servlets to run in Netscape Enterprise, Microsoft Internet Information Server (IIS), and Apache servers.

LAST UPDATED: 04 Apr 2005

Do you have something to add to this definition? Let us know.
Send your comments to techterms@whatis.com

Share - Digg This! Bookmark with Del.icio.us

RELATED CONTENT

Java
Ajax missing link in Java EE?
Java replaces XML in build process
SOA future: Sun plans to sell its own corporate culture
Eclipse tools meet database technology needs
SOA meets Web 2.0 - Where the Java EE standards fall short
Iona ties ESB to SOA management, BEA updates Kodo
Salesforce.com debuts Web services development platform
Bringing closure to Java
Eclipse versus NetBeans
James Gosling on Java, Web services and the devil

Java Server Page

SearchWebServices.com Definitions

EMAIL THIS

(Powered by WhatIs.com)

LOOK UP TECH TERMS

Powered by: Whatis.com

Search listings for thousands of IT terms:

SEARCH

Browse tech terms alphabetically:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z #

Java Server Page

definition - Java Server Page (JSP) is a technology for controlling the content or appearance of Web pages through the use of servlets, small programs that are specified in the Web page and run on the Web server to modify the Web page before it is sent to the user who requested it. Sun Microsystems, the developer of Java, also refers to the JSP technology as the Servlet application program interface (API). JSP is comparable to Microsoft's Active Server Page (ASP) technology. Whereas a Java Server Page calls a Java program that is executed by the Web server, an Active Server Page contains a script that is interpreted by a script interpreter (such as VBScript or JScript) before the page is sent to the user.

An HTML page that contains a link to a Java servlet is sometimes given the file name suffix of .JSP.

LAST UPDATED: 14 Apr 2003

Do you have something to add to this definition? Let us know.
Send your comments to techterms@whatis.com

Share - Digg This! Bookmark with Del.icio.us

RELATED CONTENT

Java
Iona ties ESB to SOA management, BEA updates Kodo
Salesforce.com debuts Web services development platform
Bringing closure to Java
Eclipse versus NetBeans
James Gosling on Java, Web services and the devil
POJO ups and downs
Web services with Flex in 30 minutes
Sun takes over JRuby
Spring and EJB 3.0 to the power of two
Ruby on Redmond?

RELATED GLOSSARY TERMS
Terms from Whatis.com – the

technology online dictionary

EmbeddedJava (SearchWebServices.com)
Java Card (SearchWebServices.com)
Java Development Kit (SearchWebServices.com)
Java Ring (SearchWebServices.com)
Java virtual machine (SearchWebServices.com)
JNDI (SearchWebServices.com)
JRun (SearchWebServices.com)
MBean (SearchWebServices.com)
PersonalJava (SearchWebServices.com)
Tomcat (SearchWebServices.com)